Activity recognition in multiple contexts for smart-house data

Kacper Sokol and Peter Flach

Intelligent Systems Laboratory, University of Bristol, UK,
{K.Sokol, Peter.Flach}@bristol.ac.uk

Abstract. Predicting human behaviour from smart-house data is an active and challenging area of research. Applications include improving quality of life and personal healthcare, e.g. smart thermostats and detecting falls. Smart-house technology is becoming increasingly popular and in many cases making a house smarter is as simple as installing off-the-shelf devices. In this paper we focus on predicting the future location of a person in a simulated smart-house environment. We consider three different occupant types who exhibit various working patterns. We develop a *versatile model* capable of adjusting to significant changes in a person's behaviour without the need of model retraining. To this end, we build a simple event calculus framework based on the Aleph Inductive Logic Programming system. Event calculus helps to handle time and persisting sensor states. Background knowledge encodes important information about the smart-house that is otherwise difficult to learn; it also facilitates transferability of the model to different house layouts. Moreover, rule models are white-box, hence human-readable. Finally, we show that a versatile model performs significantly better than other models that do not explicitly account for the context.

Keywords: location prediction, smart-house, versatile model, event calculus, Aleph

1 Introduction

We consider the problem of controlling a heating system in a living room based on behaviour and location of a resident. We want to predict a resident's location half-an-hour in advance – the time needed to preheat the room. The majority of off-the-shelf thermostats have a one-day resolution, with only two modes available: *working day* and *free day*; where each setting allows two different temperature change points: *present* and *absent* from the house. Companies such as *Nest* have built thermostats that "learn" daily routines from repetitive, manual temperature adjustments. However, such systems perform poorly on households with unstructured and irregular working patterns, which are increasingly common these days. Hence there is a need for smart thermostats [4, 7].

In most cases, smart-house data appears in the form of a time series consisting of sensor activations. The events are usually unevenly spaced through time, as they are caused by a change in sensor state. This property of time in our data causes representational difficulties well-known in logic. To overcome this issue we use *Simple Event Calculus* [9]: a logical formalisation of actions and their effects. This framework allows for *events* (sensor state changes) occurring at a single time point to activate or

terminate *fluents*: properties of a system that persist through time. Additionally, event calculus provides an approach to evaluate the model over time rather than events, which facilitates finding the best performing model for given data [11].

In this paper we consider a simple scenario of a house with two rooms, referred to as the 'toy example'. We focus on predicting not being in a selected room, `living_room`, during the next time block (as this would allow temporarily switching off the heating). To this end, we build *vanilla* and *versatile* rule-based models and show that the latter achieves top-ranked performance regardless of context of operation (working pattern) [6]. We use the *Aleph* Inductive Logic Programming system [13] to build our models.

The remaining part of this work has the following structure. In Section 2 we present related work. Section 3 gives mathematical definition of the problem. In Section 4 we describe the data that we use, together with their generation, preprocessing and feature construction steps. Section 5 presents trained models and discusses their performance. Finally, Section 6 summaries our work and explores possibilities for future research.

2 Related literature

Developing models that recognise Activities of Daily Living in a smart-house setting is a well studied subject. [14] uses a hybrid model for activity prediction in a smart-house. Part of their system is a rule-based model, used for encoding background knowledge and low-level signal extraction and analysis. Moreover, authors of [2] hand-craft a set of rules in a forward-chaining system, to produce human-readable characteristics of a smart-house. Another example of a rule-based system used in a slightly different but related setting is [5]. The authors try to predict the type of a meeting from a video recording, where multiple low-level features are extracted by a rule-based system. They define an ontology of a meeting that consists of multiple tiers, where the higher layers require predictions from the previous layers as an input. Reusing previously learnt predicates in an automatic manner as shown by [10] could greatly benefit both our work and [5]. Nevertheless, as we mainly reuse only one predicate, `context`, our work does not implement the solution proposed there.

To summarise, most work done in the area of rule-based system applications share one feature: the systems are hand-crafted. Moreover, none of them take context of operation under consideration; this may result in poor performance during deployment, if the problem setting has changed after the training data was collected. In this work we build a solution taking the best of both worlds: it uses the richness of background knowledge formulated by an expert (otherwise difficult to learn); and creates a universal model using context awareness to get top-performing location prediction.

3 Variable context time sequence completion

We assume that a smart-house fitted with n motion sensors at each time point t provides a feed of boolean (on/off) data in the form of a binary vector $\mathbf{x}_t = (x_{t1}, \ldots, x_{tn})$, where $x_{ti} \in \{0,1\}$ represents the state of a sensor i at time t. These observable events $\mathbf{x}_t$ for $t \in [1,T]$ create a sequence $[\mathbf{x}_1, \ldots, \mathbf{x}_t] \in \mathscr{X}$ – a smart-house state over time.

Furthermore, this sequence can be divided into a series of consecutive *non-observable* blocks by a series of time points $[m,\ldots,z]$: $B=[B_{1m},\ldots,B_{zT}]$ for $1<m<z<T$; where a block B_{ab} is defined as $B_{ab}=[\mathbf{x}_a,\ldots,\mathbf{x}_b]$ for $1<a<b<T$. Each block can be characterised by some unique label $L_{ab}=L(B_{ab})$ from a finite set of labels $\mathscr{L}$. In the toy example $\mathscr{L}=\{\texttt{sleep},\texttt{work},\texttt{leisure}\}$.

Additionally, a sub-sequence of events $[\mathbf{x}_r,\ldots,\mathbf{x}_s]$ for some $1\leq r<s\leq T$ occurs in an implicit, *non-observable context* $C_{rs}=C([\mathbf{x}_r,\ldots,\mathbf{x}_s])$. We assume that every complete sequence $[\mathbf{x}_1,\ldots,\mathbf{x}_T]$ is a mixture of several contexts. In our example, possible contexts are $\mathscr{C}=\{\texttt{working\_day},\texttt{working\_night},\texttt{working\_morning},\texttt{working\_afternoon},\texttt{free\_day}\}$.

Due to the above property of our data, a naïvely fitted model would implicitly depend on the sequence of contexts. Such a model is therefore ineffective, if in deployment the context is not as expected. To remedy this, we propose to use a *versatile* model that can adapt to constantly changing contexts, therefore providing accurate predictions throughout [1]. Example 2 shows a versatile rule. It predicts a person being absent in the living room in the next time block if the person is currently not in the living room, the current time block (hour of the day) is 6 and the current context is a working day. In this case the context is defined as an additional feature.

Finally, we define the binary target variable to be $y_t\in\{0,1\}$ – where $[y_1,\ldots,y_T]\in\mathscr{Y}$; and y_t indicates not being in a selected room (`living_room`) in the next $t+\delta$ time block, for $\delta>0$. In the toy example we fix $\delta=1$. With the input space $\mathscr{X}$ as defined above, our data are represented as $\mathbf{D}\subseteq\mathscr{X}\times\mathscr{Y}$ and we want to learn a function $f:\mathscr{X}\to\mathscr{Y}$ describing these data.

In the smart-house scenario presented above the versatile model output $y_{t+1}=f([\mathbf{x}_1,\ldots,\mathbf{x}_t])=\texttt{versatile\_model}(C_{tt},[\mathbf{x}_1,\ldots,\mathbf{x}_t])$, which is a function of partially observed data sequence $[\mathbf{x}_1,\ldots,\mathbf{x}_t]$ and the current context C_{tt}. As we observe larger parts of this sequence, our predictions should improve given that we have correctly identified the context. This model requires two predictions: the current context C_{tt}, dependent on the block structure in the data; and the label y_{t+1}. As the latter is dependent on the former it has to be learnt first, as a separate task. Once the `context` predicate is in place, it is used together with the partial observations $[\mathbf{x}_1,\ldots,\mathbf{x}_t]$ to predict the label y_{t+1}. The two learning tasks are separate: they access the same information (background knowledge and features), but different examples are provided for each task: the first one uses the day type, the latter requires the future location of a subject.

For the problem defined above we want to show that a vanilla model performs quite well on the data whenever the implicit context does not change. However, once it changes the vanilla model's performance drops significantly, while a versatile model performance is resistant to such variations.

4 Data

In this work, we build and evaluate our models using artificially generated data. This choice is motivated by a wide range of issues – not directly related to the variable context time series completion – that we experienced while working on real-life data sets collected by the CASAS group [3]. These issues include incomplete, missing and

noisy data, as well as labelling inconsistencies and errors due to human annotators. Using artificially generated data allows us to focus on the development of context-aware rule-based location prediction system rather than addressing real-life data issues. We use the toy example as a development test-bed, and plan to address more complex smart-house simulations and real-life data in our future research.

To generate the datasets necessary for this study, we implemented a highly customisable, stochastic smart-house simulator. The simulator takes house layout, sensors location and configuration, and high-level description of actions for the simulated agent as an input. This design guarantees each simulation to produce unique data for a strictly defined smart-house setting [12]. Therefore, every realisation of the data for a given working pattern results in slightly varying pattern from the same distribution. We generate multiple realisations of a week of data (Monday till Sunday), given each of the three working patterns: normal (full-time), part-time and shifts work.

The toy example house has two rooms: `living_room` and `bedroom`, each one fitted with a single motion sensor. Moreover, rather than using 24-hour day we discretise each one into 12 distinctive time blocks. Then a week of data is generated for each of the three working patterns. The data described here are simple to interpret and small enough to facilitate quick model construction and evaluation, yet they exhibit all the necessary characteristics of the variable context time sequence completion problem.

Spatio-temporal data generated by a smart-house pose representation issues, mainly due to the time variable being unbounded and sensor events occurring in a single time point, but causing a change persisting through time. These properties lead naturally to the use of *Simple Event Calculus*, which addresses all of the above issues [8]. Therefore, the data have to be formatted using event calculus syntax, as well as being preprocessed for the Aleph framework. The raw data i.e. sensor activations are interpreted as *events* affecting the state of *fluents* encoded with `holdsAt` predicate, which evaluates to true for each sensor during the period between its activation and deactivation. Additionally, Aleph's background knowledge encodes bindings between sensors and their room locations as well as room interconnections. Positive and negative examples are generated alongside the raw data during simulation, what guarantees perfect labelling accuracy.

Rules extracting features from the raw sensor state data are the most important building block of any rule-based system. Most of our features are location-based: either current or one of the previous locations, e.g. being present or absent in a room, visiting a room in a fixed length sliding time window or a sequence of visited rooms. Time-based features extract useful information from a UNIX timestamp e.g. date, time, time of the day, day of the week and season of the year.

Finally, we use five different contexts: `working_day`, `working_night`, `working_morning`, `working_afternoon` and `free_day`. They indicate different types of a day, mixture of which creates one of the three different working patterns. In the **normal** (*full-time*) pattern a typical working day consists of basic morning activities, 8 hours of work, followed by a leisure time and 8 hours of sleep. A typical free day consists mainly of leisure activities and having a night out. In the **part-time** pattern the free days are exactly the same as above, but working days are either *working mornings* or *working afternoons* where the work period is reduced to 4 hours and the remaining 4 hours are allocated with leisure activities. In the **shifts** pattern the free days are again the

Toy data:	*Rules:*	Normal	Shift	Part	Merged	Versatile	Majority
	Normal	**89.29**	72.62	76.19	90.48	*96.43*	75.00
	Shift	80.95	**82.14**	67.86	89.29	*90.48*	66.67
	Part	80.95	77.38	**85.71**	*97.62*	88.10	63.10
	Merged	83.73	77.38	76.59	***92.46***	91.67	68.25
	Shuffled	83.73	77.38	75.40	74.21	*84.76*	68.25

Table 1. Accuracies (in %) averaged over 3 realisations of given dataset for *vanilla* (normal, shift, part and merged), *versatile*, and *majority class* approaches for the toy dataset. Rows are data and columns are rule sets. *Normal*, *shift* and *part* correspond to data generated in particular context. *Merged* is a dataset constructed by combining 1 week of data from each context. *Shuffled* is *merged* dataset with randomly reordered days from each context. Italic numbers indicate best rule set for given dataset; bold figures indicate training set accuracies.

same as above, however they can appear during the weekdays. Additionally, there are two types of work patterns: *working days* which are exactly the same as in the normal working schedule and *working nights* – mirror image of working days.

5 Results

The main goal of our work is to show the importance of versatile models in complex scenarios like smart-house location and activity prediction. Non-versatile models learn and use implicit information about the house layout, number of residents and lifestyle patterns. A model performing well on one person's house is not guaranteed to have similar performance on a different house. A versatile model can handle multiple contexts without retraining – features in such models need to generalise well by using *markers* specific to given routines rather than fixed time points.

A major advantage of ILP rule-based systems is the white-box nature of the produced model. This gives the user a chance to understand his behaviour by simply inspecting the model, or tweak it whenever necessary by changing the background knowledge. If a person moves from one house to another and does not change living habits, a model once learnt can be reused in the new environment by simply changing sensors bindings in the background knowledge file. Additionally, background knowledge can be used to "inform" the model about the learning problem properties, which with any other learning system would have to be inferred first.

Below we present two model types: *vanilla* and *versatile*; we also compare and contrast them against *majority class classification*. For comparison, we use data for all 3 working patterns as well as *merged*: 3-weeks long data where each week comes from a different working pattern. Additionally, we use 3-weeks long data where number of days from every context is the same as in *merged* data but the order is shuffled – we call it *shuffled*. Results for the toy example are presented in Table 1; they give accuracies (in %) averaged over 3 different realisations of given working pattern under the same distribution (the agent script and the house layout).

A **vanilla** (non-versatile) model does not use latent contextual information encoded in the data. Each of *Normal*, *Shift* and *Part* models were trained on a single context data; *Merged* rule list was train with the data containing all 3 contexts (see *merged* data description).

Given our features, the most basic vanilla model learnt by Aleph is highly dependent on the *time* structure of the series (Example 1). Due to the repetitive patterns in the data, the model memorises what happened during each *day of the week* at given *time of the day* therefore performing well on the data from the same working pattern distribution. However, this model does not perform equally well when working patterns change e.g. Monday for the normal working person is no longer a working day but it is a free day.

Example 1. Example rule used by the *vanilla* model.

```
not_visiting_room_in_the_next_time_block(living_room,A):-
    holdsAt(day_number(1),A),
    holdsAt(time_block(3),A),
    holdsAt(in_room(bedroom),A).
```

A **versatile** model identifies patterns and parametrises them based on the context. Most of the patterns in our data are variations in time structure of the same events. Therefore, such model learns structure of sequences in given context, rather than its detailed dependence on time. The versatile model always outperforms self-learnt rules for each of the 3 working patterns. Additionally, it performs very close to the *Vanilla* model on *merged* data and it does not suffer significant loss in accuracy for the shuffled data. This 7% drop in performance is mostly caused by the `context` rule not always correctly recognising the type of the day from the partial data it gets.

Example 2. Example rule used by the *versatile* model.

```
not_visiting_room_in_the_next_time_block(living_room,A):-
    context(working_day,A),
    holdsAt(time_block(6),A),
    holdsAt(not_in_room(living_room),A).
```

Wilcoxon signed-rank test comparing the versatile with merged and self-learnt models for the toy dataset shows that the versatile model significantly outperforms the other two. The test rejects the null-hypothesis that the results come from the same distribution with p-values 10^{-18} for versatile vs. merged, and 10^{-52} for versatile vs. self-learnt.

6 Conclusions and future work

The work presented in this paper shows the importance of (often implicit) context in which our data is collected. Moreover, we showed that context is not always a monolithic object: it can usually be divided into smaller entities. Furthermore, we demonstrated the utility of the event calculus framework in logic when handling spatio-temporal data, and that it can be implemented in Aleph. We also presented the design of a simple versatile model. Finally, we showed that by recognising context, the model does not suffer significant loss in performance when the operating context changes, therefore avoiding time and resources expensive retraining phase.

In future work we plan to study the variable context time sequence completion problem in more detail using real-life datasets. While working with real data we aim to address the issues identified and described in Section 4. This will enable us to evaluate both vanilla and versatile models on these large scale datasets and further investigate the advantages of our approach. Last but not least, we will compare our rule-based approach with any state-of-the-art propositional learner.

References

1. Al-Otaibi, R., Prudêncio, R.B., Kull, M., Flach, P.: Versatile decision trees for learning over multiple contexts. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 184–199. Springer (2015)
2. Baryannis, G., Woznowski, P., Antoniou, G.: Rule-based real-time adl recognition in a smart home environment. In: Rule Technologies. Research, Tools, and Applications: 10th International Symposium. pp. 325–340. Springer (2016), http://dx.doi.org/10.1007/978-3-319-42019-6_21
3. Cook, D.J., Schmitter-Edgecombe, M.: Assessing the quality of activities in a smart environment. Methods of information in medicine 48(5), 480 (2009)
4. Diethe, T., Twomey, N., Flach, P.: Active transfer learning for activity recognition. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (2016)
5. Hakeem, A., Shah, M.: Ontology and taxonomy collaborated framework for meeting classification. In: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. vol. 4, pp. 219–222. IEEE (2004)
6. Hernndez-Orallo, J., Martinez-Uso, A., B.C. Prudencio, R., Kull, M., Flach, P., Ahmed, C., Lachiche, N.: Reframing in context: A systematic approach for model reuse in machine learning. AI Communications 29(5), 551–566 (11 2016)
7. Kafalı, Ö., Romero, A.E., Stathis, K.: Activity recognition for an agent-oriented personal health system. In: International Conference on Principles and Practice of Multi-Agent Systems. pp. 254–269. Springer (2014)
8. Katzouris, N., Artikis, A., Paliouras, G.: Incremental learning of event definitions with inductive logic programming. Machine Learning 100(2-3), 555–585 (2015)
9. Kowalski, R., Sergot, M.: A logic-based calculus of events. In: Foundations of knowledge base management, pp. 23–55. Springer (1989)
10. Lin, D., Dechter, E., Ellis, K., Tenenbaum, J.B., Muggleton, S.H.: Bias reformulation for one-shot function induction (2014)
11. Shanahan, M.: The event calculus explained. In: Artificial intelligence today. Springer (1999)
12. Sokol, K.: Shgen: v1.0 (Dec 2015), http://dx.doi.org/10.5281/zenodo.34710
13. Srinivasan, A.: The Aleph manual, http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/aleph
14. Storf, H., Becker, M., Riedl, M.: Rule-based activity recognition framework: Challenges, technique and learning. In: 2009 3rd International Conference on Pervasive Computing Technologies for Healthcare. pp. 1–7. IEEE (2009)